R E M A R K S

Applicant has carefully considered the Office Action of December 21, 2007, rejecting all of the claims. Applicant wishes to express his appreciation to the Examiner for the interview conducted on October 16, 2007 by the undersigned.

The present response is intended to implement the conclusions of the interview and be fully responsive to all points of objection and rejection raised by the Examiner, and is believed to place the application in condition for allowance. Favorable reconsideration and allowance of the application are respectfully requested.

Claims **1, 5, 17, 21** and **30** have been amended. No claims have been deleted. Therefore, claims **1-30** remain in the case.

The present invention comprises a combination of three major elements:

1) visualizable computer executable modeling language for the definition of software solutions;
2) a modeling environment for visually defining the software solutions in the modeling language; and
3) a runtime engine that executes solutions defined in the modeling language.

As stated at the end of paragraph 0082 of the published application, the method of the present invention effectively achieves elimination of the writing of source code for developing software applications.

The Examiner has rejected claims 21-23 and 30 under 35 U.S.C. 101 as being directed to non-statutory subject matter. The Examiner has also stated that claims 21 and 30 are software per se.

It is believed that the Examiner's rejection stems from the lack of hardware elements, making these claims intangible.

Thus, claim 21 has been amended as a Beauregard claim drawn to a computer program product.

Since claims 22 and 23 of the present invention are dependent on claim 21, which is deemed patentable, no further

amendment is Claim 30 was previously added as a Beauregard claim drawn to a computer program product and has now been amended with the same format as amended claim 1. Thus, no further amendment is needed for claim 30.

As per the Examiner's objection as expressed in the interview, claim 5 has been amended to delete the step which has previously been incorporated in claim 1, the step needed for claims 22-23.

of "defining a portion of said slots as having one of the following sub-classifications: mandatory; and optional;".

It is the Applicant's position that the present amendments to claims 21 and 30 provide inventive claim formulations of statutory subject matter, so that the Sec. 101 rejection is improper, and Applicant respectfully requests it be withdrawn.

The Examiner has rejected claims 1-30 under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. It is the Examiner's position that there is no support in the specification for the recitation in independent claims 1 and 30 "such that the need for writing code in any programming language to implement the software applications is eliminated".

The Examiner maintains that the original disclosure indicates some form of code writing may very well occur (see at least: page 1, line 8; page 4, line 14; page 4, line 17; page 4, lines 24-27; page 13, lines 5-8).

During the interview, the Examiner stated that he believes that the invention is "graphical modeling to generate/produce code." The Applicant fully agrees with the Examiner that the invention introduces graphical modeling to generate/produce code, and this embodiment is recited in claim 21.

The other embodiment, recited in claim 17, is that the visualizable modeling language sets up the execution by the runtime engine.

It is important to distinguish between a "Modeling Phase" in which a developer is defining applications through the

modeling system, and an "Execution Phase" in which the computer executes the application as defined in the Modeling Phase (either through a "Runtime Engine" or a "Code Generator").

In the Modeling Phase, the developer uses a tool with a sophisticated GUI, thus "display elements" are required. The result of this phase are the models created by the developer.

In the Execution Phase, there is no need for any GUI component. Both the Runtime Engine and the Code Generator use as input the models created in the Modeling Phase, and they don't use any "display elements" to do their work.

Apparently, the unfortunate use in the specification of the vague term "substantially" together with "overcoming the need to write computer source code" created a purely semantic misunderstanding of what the invention actually does.

To assist the Examiner in better understanding the invention, thus correcting the misunderstanding, his attention is drawn to the following text portions (emphasis added):

In paragraph [0116], describing Fig. 3: "Flows are graphically represented by arrows."

In paragraphs [0337] and [0338]: "The modeling environment comprises three main components: a Graphical User Interface (GUI) tool, or "Modeling Tool," which enables a user to visually create models, which are modeling language definitions of the components of the developed solutions".

In paragraph [0341]: "The workspace of the modeling tool..... for displaying..... graphical representations of corresponding hierarchies of modeling language models."

In paragraph [0344]: "Users of the modeling tool create and edit modeling language models using various GUI operations such as creating new process or data models through menu operations, adding components to process or data models by dragging models from palettes of existing models, modifying attributes of models and of model components, etc. The modeling

tool prevents the user from creating models that are inconsistent with the restrictions of the modeling language."

In paragraph [0352]: "Employing this intuitive graphical interface for further zooming into sub models of sub models allows the modeler an "infinite zooming capability" into any detail of any model.

In paragraph [0380]: "... the modeling tool can be used as a "visual debugger" to trace the execution of models by the runtime engine....graphical means are used to give the user full information on the full status of each instance. The user can trace the execution step by step,.."

Thus, since the definition of the invention includes "graphical modeling to generate/produce code", which is thoroughly described and illustrated in the application, and since there is nothing in the application that can be construed as "writing code in any programming language," the term "substantially" is simply a semantic problem, whether this term is used or not. Applicant has chosen to delete this term.

Therefore, by careful reading of the application and by complete removal of the term "substantially," it can be seen that the inventive subject matter was already described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventors, at the time the application was filed, had possession of the claimed invention.

Further, the visual modeling, which Applicant lays claim to, is not simply programming in a particular modeling language and producing code in that language, since nowhere is any form of code written. Therefore, there is support for not writing code in "any programming language" (see spec. para. [0019]). Thus, all the independent claims (and thus the dependent claims) are supported in claiming no need to write code.

The Examiner stated in the interview that as he sees it, Applicant goes from models to code directly, and that the claims should use positive recitation. Claim 1 has been so amended, as above, with support from the specification, and at

the end of the claim, the wording "without further coding" is used as per spec. para. [0019] to highlight the differences.

The Examiner stated in the interview that the present invention looks like a different invention from the prior art, and that he agrees that if Applicant is correct and the developer is not writing code (per claim 1 at end), then this feature is inventive.

Further support for Applicant's position that the invention does not require writing code is found in paragraph 17 as amended, and paragraphs 19 and 21 of the specification (emphasis added):

[0017] It is still a further object of the present invention that instead of writing code, **a model is created by the developer, wherein the model includes everything needed for defining the application in adequately precise terms.**

[0019] It is one further object of the present invention that no further coding is required once the solution is visually defined by the modeling language, wherein **the [modeling] language is rich enough and precise enough for a computer to execute an application model that is created using a computer with the appropriate graphic user interface** (GUI).

[0021] It is one more object of the present invention that **the modeling language is sufficiently intuitive so that it can be understood by humans in a visual manner.**

It is the Applicant's position that the present amendments to the specification and claims overcome the basis of the 35 U.S.C. 112, first paragraph rejection, and Applicant respectfully requests that it be withdrawn.

The Examiner has rejected claims 1-12 and 14-16 and 30 under 35 U.S.C. 102(b) as being anticipated by Williams (US Patent No. 5,850,548), relative to column 2, lines 20-39, disclosing a visual programming environment based on a high level hierarchical data flow model.

Williams discloses that components communicate by sending and receiving "messages" on their "ports." Components can be constructed with a conventional programming language, such as C++ or Pascal, or constructed with Monet programs constructed entirely out of components which communicate via connections.

Although both Williams and the present invention are based on common dataflow notions and constructs, the patentable parts of each are the specific characteristics of each language and the development process defined for building software applications.

The following are some major differences between the present invention and Williams:

In claim **1** of Williams, a "program component," corresponding to the "process model" of the present invention, has properties, some of which "surface" as "port connections," corresponding to the "slots" of the present invention.

By contrast, in the present invention a slot is <u>not</u> a property of a process model as in Williams, but rather a sub-component of a process model.

In Williams, connecting ports graphically is a mandatory step (step (e) in claim 1). By contrast, in the present invention "flow rules" are optional components.

In Williams, the user needs to "surface" a property as a "port connection" to make it available for connection (see step (e)(i) of claim **1**). **In the present invention there is no "surfacing" step.**

The present invention includes several notions and constructs which do not exist in Williams:

Claim **1**: "Data models" (atomic and composite), which can be a sub-components of process models or other data models. Williams only describes properties of components.

Claim **1** has been amended by incorporating a portion of the language of claim **5**. The present invention system classifies input slots as mandatory or optional. This, when combined with the ability to model arbitrarily complex data structures, gives

a significant expressive power, which is lacking in other dataflow languages.

Claim **7**: Mapping between database tables and data models.

Claim **10**: Various composition methods of composite data models (concatenation, collection, selection, recurrence).

As stated in the decision in In Re Marshall, 198 USPQ 344 (1978), "To constitute an anticipation, all material elements recited in a claim must be found in one unit of prior art…". Since the Williams reference neither 1) identically describes the invention, nor 2) enables one skilled in the art to practice it, Applicant deems the 102(b) rejection improper, and respectfully requests that it be withdrawn.

The dependencies of claims **2-12** and **14-16**, ultimately relating to claim 1, result in no amendments being needed.

The Examiner has rejected claim **13** under 35 U.S.C. 103(a) as being unpatentable over Williams, referring to column 2, lines 20-39, and column 5, line 31 to column 13, line 42. Again, since claim **13** of the present invention is indirectly dependent on claim **1** no further amendment is needed for claim **13**.

The Examiner has rejected claims **17-20, 24-25** and **27-28** under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Parr et al, (US20020095653).

Parr discloses a visual architecture software language to define discrete software modules and data flow and control flow between them. It includes a code generation step (called "compilation") to convert the visual representation into executable code in the form of executable instructions and data tables.

Code generation is a known technique to convert the definition of a software component (e.g. a software application) from one representation to another, typically from a high-level, user-friendly notation to a lower-level notation.

Although both Parr and the present invention use code generation, the patentable parts of each are the specific

characteristics of each language and the code generation algorithm used to generate code from it.

The code generator of the present invention is inherently related to the visual modeling language of the present invention, and it has to create code that is functionally 100% equivalent to the operation performed by the runtime engine described in claim 17 of the present invention. None of the features of this complex code generator can be inferred, logically or practically, from the mere notion of "compilation" mentioned by Parr.

Note that Parr discusses code generation in very general terms without explaining what "compilation" means (Parr's claim 16 is just half a sentence that says nothing on the generated code: "...compiling the visual representation to generate executable code, said executable code comprising").

The Examiner has rejected claims **21 - 23** under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Goodwin et al (US 6,199,195).

Goodwin deals with "automatically generated object-oriented source code." This is not the case with the code generator of the present invention.

Furthermore, the essence of the present invention is the completeness and preciseness of the language, which enable the needed code generation.

Although both Goodwin and the present invention use code generation, the patentable parts of each are the specific characteristics of each language and the code generation algorithm used to generate code from it.

Therefore, the code generator of the present invention is inherently related to the language of the present invention.

The Examiner has rejected claims **26** and **29** under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Parr et al, in further view of Goodwin et al.

It is respectfully put forward by the Applicant that there is no reason to consider the combination of Williams, Parr and

Goodwin as rendering the present invention unpatentable, since they include no provision for eliminating the need for writing code in any programming language to implement software applications.

It is the Applicant's position that the combination of the Williams, Parr and Goodwin references to form the basis of the Sec. 103(a) rejection is improper, and Applicant respectfully requests that it be withdrawn.

Therefore, claims **1 - 30** are deemed to be patentable.

LEGAL ARGUMENTS

With regard to the combination proposed by the Examiner, the legal standard for this combination has not been met by the Examiner. The sum of the previously cited legal precedents set forth by In Re Lintner, In Re Regel, In Re Clinton, Application of Wesslau, Grain Processing Corp., and In Re Dance, all support the requirement that before prior art references can be combined or modified, there must be some suggestion or motivation found in the art to make the combination or modification.

The Examiner is relying on speculation and hindsight reconstruction of the references in view of the invention, and the Examiner is using an arbitrary combination of references.

The only motivation for the combination suggested by the Examiner is provided by the Applicant's invention, which utilizes a visual modeling language to eliminate the need for writing code in any programming language to implement software applications.

Applicant maintains that there is no reason to consider the prior art references either individually or in combination, as rendering the invention obvious.

Based on the amendments to the claims and the above remarks, Applicant believes that no new issues are raised and the invention is novel and inventive and that all the pending claims are deemed to be allowable. In view of the foregoing remarks, all of the claims in the application are deemed to be

allowable. Further reconsideration and allowance of the application is respectfully requested at an early date.

Respectfully submitted,

Edward Langer, Pat. Atty.
Attorney for Applicant
Reg. No. 30, 564

Shiboleth, Yisraeli, Roberts and Zisman LLP
1 Penn Plaza, Suite 2527, New York, NY 10119
Tel.: 212-244-4111 Fax.: 212-563-7108
466988